PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-091437
(43)Date of publication of application : 27.03.2002

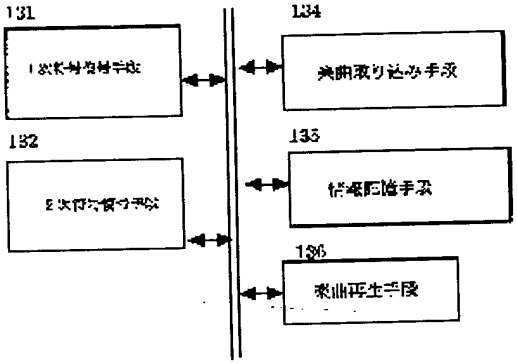
(51)Int.Cl. G10H 1/00
G06F 5/00
G10K 15/02
H03M 7/30
H04M 1/00

(21)Application number : 2000-283179
(22)Date of filing : 19.09.2000

(71)Applicant : VICTOR CO OF JAPAN LTD
(72)Inventor : HIKAWA KAZUO

(54) PERFORMANCE INFORMATION COMPRESSING DEVICE AND PERFORMANCE INFORMATION DECODING DEVICE AND TELEPHONE TERMINAL EQUIPMENT

(57)Abstract:
PROBLEM TO BE SOLVED: To efficiently compress and store the data quantity of an incoming melody in a memory, and to start the reproduction of the incoming melody immediately after call incoming.
SOLUTION: The performance information of a piece of music to be reproduced at the time of call incoming is divided into not less than two blocks on a time base, and the first block is stored in a memory so as to be defrosted and performed immediately after the call incoming.



LEGAL STATUS

[Date of request for examination] 28.03.2003
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－91437
（P2002－91437A）
(43)公開日　平成14年３月27日(2002.3.27)

(51)Int.Cl.'	識別記号	FI		テーマコード'(参考)
Ｇ１０Ｈ　1/00		Ｇ１０Ｈ　1/00	Ｚ	５Ｄ３７８
	１０２		１０２Ｚ	５Ｊ０６４
Ｇ０６Ｆ　5/00		Ｇ０６Ｆ　5/00	Ｈ	５Ｋ０２７
Ｇ１０Ｋ　15/02		Ｇ１０Ｋ　15/02		
Ｈ０３Ｍ　7/30		Ｈ０３Ｍ　7/30	Ｚ	

審査請求　未請求　請求項の数10　ＯＬ　（全 21 頁）　最終頁に続く

(21)出願番号　特願2000－283179(P2000－283179)

(22)出願日　平成12年９月19日(2000.9.19)

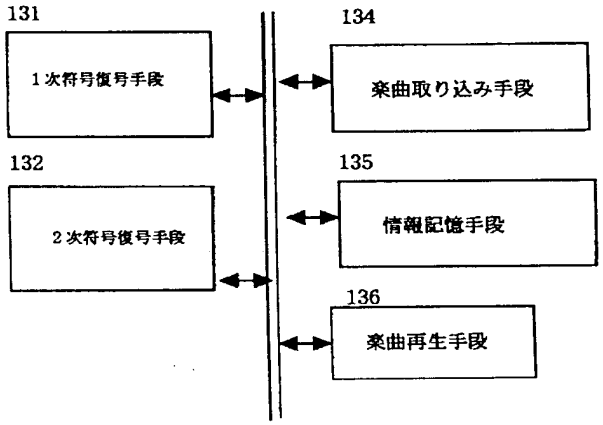
(71)出願人　000004329
日本ビクター株式会社
神奈川県横浜市神奈川区守屋町３丁目12番地

(72)発明者　飛河　和生
神奈川県横浜市神奈川区守屋町３丁目12番地　日本ビクター株式会社内

Ｆターム(参考)　5D378 MM28 MM62 MM65 MM68 MM96 QQ02 QQ06 QQ23 QQ24 QQ27 QQ38
5J064 AA02 BA15 BC01 BC02 BD02 BD03
5K027 AA11 FF03 FF25

(54)【発明の名称】　演奏情報圧縮装置、演奏情報復号装置及び電話端末装置

(57)【要約】
【課題】　着信メロディのデータ量を効率的に圧縮してメモリに記憶すると共に、通話着信後直ちに着信メロディの再生を開始する。
【解決手段】　着信時に再生する楽曲の演奏情報を時間軸上で２つ以上のブロックに分割し、最初のブロックについては着信後すぐに解凍して演奏できる状態でメモリに記憶しておく。



1

【特許請求の範囲】

【請求項１】演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報とに分離し、前記各情報をそれぞれ独立した領域に配置した１次符号を生成する１次符号生成手段と、

前記１次符号生成手段により生成された１次符号の各領域の情報を圧縮する２次符号生成手段と、

前記演奏情報を時間軸上で２つ以上のブロックに分割し、少なくとも最初のブロックについては前記１次符号生成手段による圧縮及び／又は前記２次符号生成手段による圧縮を行わないことを特徴とする演奏情報圧縮装置。

【請求項２】前記１次符号生成手段は、各イベント間の相対時間の公約数、及び、各音符の長さの公約数を算出し、各イベント間の相対時間、及び、各音符の長さの値をそれぞれの公約数で除算した後符号化することを特徴とする請求項１に記載の演奏情報圧縮装置。

【請求項３】前記１次符号生成手段は、各音符の音程情報をそれ以前に出現した音符の音程の数値を使って一定の関数式に従って算出した数値と実際の音程の数値との残差で表すことを特徴とする請求項１又は２に記載の演奏情報圧縮装置。

【請求項４】前記１次符号生成手段は、各音符の強さ情報をそれ以前に出現した音符の強さの数値を使って一定の関数式に従って算出した数値と実際の強さの数値との残差で表すことを特徴とする請求項１乃至３のいずれか一項に記載の演奏情報圧縮装置。

【請求項５】前記１次符号生成手段は、特定の種類のイベントのパラメータ値をそれ以前に出現した同種類のイベントのパラメータ値を使って一定の関数式に従って算出した数値と実際のパラメータ値との残差で表すことを特徴とする請求項１乃至４のいずれか一項に記載の演奏情報圧縮装置。

【請求項６】前記１次符号生成手段は、前記各情報を当該領域においてトラック順に配置したことを特徴とする請求項１乃至５のいずれか一項に記載の演奏情報圧縮装置。

【請求項７】前記１次符号生成手段は、前記各領域をデータの性質が似ている領域同志が近くなるように配置したことを特徴とする請求項６に記載の演奏情報圧縮装置。

【請求項８】演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報とに分離し、前記各情報をそれぞれ独立した領域に配置した１次符号を復号する演奏情報復号装置であって、供給される前記１次符号を演奏情報に復号する１次符号復号手段を有することを特徴とする演奏情報復号装置。

【請求項９】演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、

2

前記演奏情報を時間軸上で二つ以上のブロックに分割し、少なくとも最初のブロックが電話着信時に直ちに演奏できるように二番目以降のブロックのみを圧縮して記録することを特徴とする電話端末装置。

【請求項１０】演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、

前記演奏情報を時間軸上で二つ以上のブロックに分割して、それぞれのブロックを圧縮するときに、

少なくとも最初のブロックは、電話着信時に直ちに演奏でき、且つ、圧縮された二番目のブロックの復号を行うことができる時間の長さ及び／又は二番目以降のブロックの圧縮方法よりも短時間で復号可能な圧縮方法で圧縮されていることを特徴とする電話端末装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、演奏情報のデータ量を圧縮する演奏情報圧縮装置、復号を行う演奏情報復号装置及びこれらの装置を内蔵した電話端末装置に関する。

【０００２】

【従来の技術】一般に、ＭＩＤＩ（Musical Instrument Digital Interface）データを保存する方式として、スタンダードＭＩＤＩファイル（以下、ＳＭＦという）が広く用いられ、このＳＭＦでは図１９に示すように個々の演奏情報がデルタ（Δ）タイムとイベントの２つの要素により構成されている。Δタイムは、隣合ったイベント間の相対時間を表し、イベントはノートオン又はノートオフのステータス、音程（ノートナンバ）や音の強さ（ベロシティ）などの種々の演奏情報を含む。ここで、演奏情報とは音符により示される音程、音の長さの他に、音の強さ、楽曲の演奏上の拍子、テンポなどの情報、さらに音源の種類、リセットコントロールの情報などを含むものを指すものとする。また、このＳＭＦでは各演奏情報が時間順に並んでトラックを構成している。ここで、ＳＭＦ方式は記憶容量や伝送路の効率的利用という点ではあまり優れたものではないので、携帯電話の着信メロディや通信カラオケ、音楽データベースのように多数の楽曲データを記録、伝送するシステムでは、演奏情報のデータ量を効率的に圧縮することが求められている。

【０００３】これと同時に、携帯電話の着信メロディでは、電話が着信したときに着信したことを着信メロディの再生によって知らせるが、この着信メロディのファイルをＳＭＦファイルとして携帯電話本体に保存する際に、携帯電話本体の記憶容量の小さいメモリー部分に、できる限り多くの曲ファイルを記録することが求められている。

【０００４】一方、テキスト等のデジタルデータを可逆的（ロスレス）に圧縮する方法としては、文字列（デー

タパターン）が繰り返すことを利用して繰り返し分を圧縮するＬＺ（Lempel-Zif）法が現在広く使用され、ＬＺ法は一般に使われている可逆式圧縮方法の中で、圧縮率が最も高いと言われている。ＬＺ法は文字列が繰り返すことを利用して圧縮するので、入力データの中の限られた範囲内に出現する同一のデータパターンの数が多く、且つ同一データパターン長が長い場合に圧縮率が高くなるという性質を有する。

【０００５】そこで、本出願人は、特開平９？１６１６８において、ＬＺ法により圧縮する前に予め、同一のデータパターンの長さが長く、出現回数が多く且つ近い距離で出現するように、演奏情報を音程と、強さと、長さとその他の情報に分離し、それぞれ独立した領域に配置するように、演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報とその他の情報に分離し、各情報をそれぞれ独立した領域に配置した１次符号を生成する１次符号生成手段と、前記１次符号生成手段により生成された１次符号の各領域の情報をＬＺ法により圧縮する２次符号生成手段とを有する演奏情報圧縮装置を提案した。

【０００６】この装置では、１次符号生成手段が、各イベント間の相対時間の公約数及び各音符の長さの公約数を算出し、各イベント間の相対時間及び各音符の長さの値をこれらの公約数で除算した後符号化するよう構成されていることは前記提案の好ましい態様である。また、１次符号生成手段が、各音符の音程情報をそれ以前に出現した音符の音程の数値を使って一定の関数式に従って算出した数値と実際の音程の数値との残差で表すよう構成されていることは本発明の好ましい態様である。また、１次符号生成手段が、各音符の強さ情報をそれ以前に出現した音符の強さの数値を使って一定の関数式に従って算出した数値と実際の強さの数値との残差で表すよう構成されていることは前記提案の好ましい態様である。

【０００７】また、１次符号生成手段が、特定の種類のイベントのパラメータ値をそれ以前に出現した同種類のイベントのパラメータ値を使って一定の関数式に従って算出した数値と実際のパラメータ値との残差で表すよう構成されていることは前記提案の好ましい態様である。また、１次符号生成手段が、各情報を当該領域においてトラック順に配置することは本発明の好ましい態様である。さらに、１次符号生成手段が、前記各領域をデータの性質が似ている領域同志が近くなるように配置することは前記提案の好ましい態様である。

【０００８】

【発明が解決しようとする課題】最近の着信メロディはｉモード等のサービスサイトよりダウンロードすることによって着信メロディファイルを得ることが主流になりつつあるが、ユーザは着信メロディ楽曲ファイル（以下ファイルと称する）の入手のために会費を支払っており、携帯電話本体の内蔵メモリが少ないと、既にダウンロードしたファイルのファイルサイズの合計がメモリーの容量を超えた際に、以前に購入したファイルを消去しなければならなくなる。また、この状況はユーザが自ら演奏データを入力する場合、あるいは専用のパーソナルコンピュータ等のソフトウエアを用いてファイルを携帯電話本体に記録する場合に於いても、同様な問題が発生する。また、今後は着信メロディのファイルフォーマットとしてＳＭＦ（スタンダードＭＩＤＩファイル）が用いられることが予想され、従来の専用ファイルよりも個々のファイルサイズが大きくなってしまう点が問題点として指摘されている。

【０００９】そこで、前記提案に記したようなファイル圧縮技術を用いることにより、ファイルサイズを小さくすることが容易に考えられるが、携帯電話に搭載されるＣＰＵによって前記提案のような圧縮処理を行うと、再生しようとするファイルの全てを解凍し終わらない限り演奏の開始が行えないことと、実際の端末側の処理速度が遅いことにより、電話を着信してから着信メロディの再生音を出力することが可能となるまでに、かなりの時間がかかってしまうことが問題となっている。

【００１０】本発明は上記従来の問題点に鑑み、そこで、予め携帯電話などの電話端末の内部・外部メモリーなどにあらかじめ指定された１つまたは複数のファイルを解凍した状態で保存することにより、電話着信時に即座に着信メロディを再生することを可能とする。

【００１１】

【課題を解決するための手段】本発明は上記目的を達成するために、演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報に分離し、前記各情報をそれぞれ独立した領域に配置した１次符号を生成する１次符号生成手段と、前記１次符号生成手段により生成された１次符号の各領域の情報を圧縮する２次符号生成手段と、前記演奏情報を時間軸上で２つ以上のブロックに分割し、少なくとも最初のブロックについては前記１次符号生成手段による圧縮及び／又は前記２次符号生成手段による圧縮を行わないことを特徴とする演奏情報圧縮装置を提供する。

【００１２】また、前記１次符号生成手段は、各イベント間の相対時間の公約数及び各音符の長さの公約数を算出し、各イベント間の相対時間及び各音符の長さの値をこれらの公約数で除算した後符号化することを特徴とする請求項１に記載の演奏情報圧縮装置を提供する。

【００１３】更に、前記１次符号生成手段は、各音符の音程情報をそれ以前に出現した音符の音程の数値を使って一定の関数式に従って算出した数値と実際の音程の数値との残差で表すことを特徴とする請求項１又は２に記載の演奏情報圧縮装置を提供する。

【００１４】更にまた、前記１次符号生成手段は、各音符の強さ情報をそれ以前に出現した音符の強さの数値を

使って一定の関数式に従って算出した数値と実際の強さの数値との残差で表すことを特徴とする請求項１乃至３のいずれか一項に記載の演奏情報圧縮装置を提供する。

【００１５】また更に、前記１次符号生成手段は、特定の種類のイベントのパラメータ値をそれ以前に出現した同種類のイベントのパラメータ値を使って一定の関数式に従って算出した数値と実際のパラメータ値との残差で表すことを特徴とする請求項１乃至４のいずれか一項に記載の演奏情報圧縮装置を提供する。

【００１６】また、前記１次符号生成手段は、前記各情報を当該領域においてトラック順に配置したことを特徴とする請求項１乃至５のいずれか一項に記載の演奏情報圧縮装置を提供する。

【００１７】更に、前記１次符号生成手段は、前記各領域をデータの性質が似ている領域同志が近くなるように配置したことを特徴とする請求項６に記載の演奏情報圧縮装置を提供する。

【００１８】更にまた、演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報に分離し、前記各情報をそれぞれ独立した領域に配置した１次符号を復号する演奏情報復号装置であって、供給される前記１次符号を演奏情報に復号する１次符号復号手段を有することを特徴とする演奏情報復号装置を提供する。

【００１９】また更に、演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、前記演奏情報を時間軸上で二つ以上のブロックに分割し、少なくとも最初のブロックが電話着信時に直ちに演奏できるように二番目以降のブロックのみを圧縮して記録することを特徴とする電話端末装置を提供する。

【００２０】また、演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、前記演奏情報を時間軸上で二つ以上のブロックに分割して、それぞれのブロックを圧縮するときに、少なくとも最初のブロックは、電話着信時に直ちに演奏でき、且つ、圧縮された二番目のブロックの復号を行うことができる時間の長さ及び／又は二番目以降のブロックの圧縮方法よりも短時間で復号可能な圧縮方法で圧縮されていることを特徴とする電話端末装置を提供する。

【００２１】

【発明の実施の形態】以下、図面を参照して本発明の実施の形態について説明する。図１は本発明に係る演奏情報圧縮装置の一例を示すブロック図、図２は図１の１次符号生成手段の一例を詳細に示すブロック図、図３は図２のチャネル分離手段により作成されるチャネルマップを示す説明図、図４は図２の解析手段の処理を説明するためのフローチャート、図５は図２の解析手段により作成されるノートテーブルを示す説明図、図６は図２の解析手段により作成されるコントローラテーブルを示す説明図、図７は音符を表現するＳＭＦのΔタイムと本実施例のデュレーションの関係を示す説明図、図８は図２のノートΔ符号生成手段の処理を説明するためのフローチャート、図９は図２のノートΔ符号生成手段により生成されるノートΔ符号を示す説明図、図１０は図２のデュレーション符号生成手段の処理を説明するためのフローチャート、図１１は図２のデュレーション符号生成手段により生成されるデュレーション符号を示す説明図、図１２は図２のノートナンバ符号生成手段により生成されるノートナンバ符号を示す説明図、図１３は図２のベロシティ符号生成手段により生成されるベロシティ符号を示す説明図、図１４は図２のコントローラ符号生成手段により生成されるコントローラ符号を示す説明図、図１５はＳＭＦの連続イベントブロックを示す説明図、図１６は本実施例の連続イベントブロックを示す説明図、図１７は図１６の連続イベントブロックの効果を示す説明図、図１８は図２の符号配置手段により並べ替えられた１次符号を示す説明図、図２０は、本発明による端末装置のうち、１次符号化された状態で指定楽曲を保存する態様による端末の構成を示す説明図、図２１は、本発明による端末装置のうち、１次符号化も２次符号化もされない状態で指定楽曲を保存する態様による端末の構成を示す説明図、図２２は、本発明による端末装置のうち、１次符号化された状態で指定楽曲を追加的に保存する態様による端末の構成を示す説明図、図２３は、本発明による端末装置のうち、１次符号化も２次符号化もされない状態で指定楽曲を追加的に保存する態様による端末の構成を示す説明図である。

【００２２】先ず、入力データ１は一例として図７、図１９に示すようなＳＭＦであり、ＳＭＦのフォーマットはΔタイム、ステータス、ノートナンバ及びベロシティにより構成されている。ここで、本明細書では「発音開始イベント」を「ノートオンイベント」、「発音停止イベント」を「ノートオフイベント」と呼ぶ。また「ノートオンイベント」と「ノートオフイベント」を合わせて「ノートイベント」と呼び、それ以外の「イベント」を「コントローライベント」と呼ぶ。

【００２３】図１において、ＳＭＦフォーマットの入力データ１は１次符号生成手段２により解析され、少なくとも演奏情報を音程と、強さと、長さとその他の情報に分離され、各情報がそれぞれ独立した領域に配置した１次符号３が生成される。この１次符号３の各領域の符号は２次符号生成手段４によりＬＺ法で圧縮され、２次符号５が生成される。なお、このように圧縮されたデータは図２０～図２８に詳しく示す復号装置によりＬＺ法で復号され、音程と、強さと、長さとその他の情報に基づいて音符が復元される。

【００２４】１次符号生成手段２は例えば図２に詳しく示すように、チャネル分離手段１１と、解析手段１２

と、ノートΔ符号生成手段１３と、コントローラΔ符号生成手段１４と、デュレーション符号生成手段１５と、ノートナンバ符号生成手段１６と、ベロシティ符号生成手段１７と、コントローラ符号生成手段１８と符号配置手段１９で構成され、この例では１つの音符を示すノートΔ符号、コントローラΔ符号、デュレーション符号、ノートナンバ符号、ベロシティ符号及びコントローラ符号の６種類の１次圧縮符号が符号配置手段１９により並べ換えられ、１次符号３として２次符号生成手段４に出力され、２次符号生成手段４により２次圧縮される。

【００２５】チェネル分離手段１１ではＳＭＦ１の１トラックに複数チャネルのイベントが含まれているか否かのチェックを行ない、複数チャネルのイベントが含まれている場合には、１トラックの中に１チャネルのイベントのみが含まれるように、トラックの分割を行う。そして、図３のようなトラックとチャネル番号の対応を表したチャネルマップを作成し、これ以降の処理はトラック単位で行う。ここで、ＳＭＦのイベントの大半はチャネル情報を含んだものであるが、このようにトラック分割とチャネルマップの作成を行うことにより、個々のイベントのチャネル情報を省略することができ、データ量を削減することができる。

【００２６】解析手段１２では図４に示す処理を行い、トラック毎に図５に示すようなノートテーブルと図６に示すようなコントローラテーブルを作成する。先ず、ＳＭＦから順次Δタイムとイベントを読み出し（ステップＳ１）、Δタイムからトラックの先頭を基準としたイベントの時間（以下では単に、イベントの時間と呼ぶ）を計算する（ステップＳ２）。次にイベントを解析し、イベントを「ノートオンイベント」、「ノートオフイベント」、「コントローライベント」の３種類に分類する。

【００２７】そして、「ノートオンイベント」の場合には図５に示すようなノートテーブルにノートナンバ（音符の音程）とベロシティ（音符の強さ）を登録し（ステップＳ３→Ｓ４）、「ノートオフイベント」の場合にはデュレーション（音符の長さ）を計算してノートテーブルに登録する（ステップＳ５→Ｓ６）。また、「コントローライベント」の場合には図６に示すようなコントローラテーブルに登録し（ステップＳ７）、このようにして演奏情報毎にノートテーブルとコントローラテーブルを作成する（ステップＳ８→Ｓ１）。

【００２８】ここで、ノートテーブルは図５に示すようにトラックのノート（音符）イベント情報を時間順に並べたものであり、コントローラテーブルは図６に示すように、トラックのコントローラ（音符以外）の情報を時間順に並べたものである。また、「ノートオンイベント」の場合にノートナンバとベロシティを書き込む際に、イベントの時間をノートテーブルの所定の欄に書き込み、また、ノートテーブルの「ノートオフ参照」欄を初期値として「０」にセットする。

【００２９】また、イベントがノートオフであれば、ノートテーブルを先頭から走査して、ノートオフイベントの時間よりも早く、かつノートナンバが同じで、かつノートオフ参照欄が「０」にセットされているノートを選び出し、対応させる。そして対応するノートオンの時間Ｔｏｎとノートオフの時間Ｔｏｆｆとの差（Ｔｏｆｆ－Ｔｏｎ）を「デュレーション（音符の長さ）」とし、ノートテーブルの「デュレーション」欄に記録するとともに、「ノートオフ参照」欄を「１」にセットする。

【００３０】ここで、「デュレーション」という概念はＳＭＦにはないが、これを用いることによりノートオフイベントを省略できるので、データ容量が削減できる。ＳＭＦにおいて、１つの音符は図７のように１つのノートオンイベントと１つのノートオフイベントの組で表され、また、ノートオフイベントの前のΔタイムがデュレーションに相当する。ノートオフイベントのノートナンバは、ノートオンイベントとの対応を取る為に必要であり、デュレーションという概念を使ってノートオンとノートオフの対応を取っておけば不要である。

【００３１】また、ノートオフイベントのベロシティは、ＭＩＤＩデータを受け取るほとんどの音源が実際にはこの値を使用しないので、削除しても問題ない。したがって、ノートオフイベント（３バイト）を省略することにより、場合によってはΔタイムのデータ量が増えることもあるが、いずれにしてもノートオフイベント省略の効果の方が大きく、１つの音符につき最大３バイト分を削減することができる。その結果、１つの楽曲の中に１万個程度の音符が含まれている場合も珍しくないので、この場合には最大３０Ｋバイトの削減ができることになり、圧縮効果が大きい。

【００３２】イベントがノートオン、ノートオフ以外のイベントであれば、イベントの時間とイベントの内容をコントローラテーブルに登録し、このようにしてノートテーブルにはＮＡ個のイベントが登録され、コントローラテーブルにはＮＢ個のイベントが登録される。

【００３３】次に、ノートΔ符号生成手段１３とコントローラΔ符号生成手段１４について説明する。この２つは、同じような処理内容であるので、以下ではノートΔ符号生成手段１３を例に取って説明する。ノートΔ符号生成手段１３は図８に示すように、先ず、前述したノートテーブルに登録された各イベントに対し、その時間Ｔ［ｉ］と１つ前のイベントの時間Ｔ［ｉ－１］との差

ΔＴ［ｉ］＝（Ｔ［ｉ］－Ｔ［ｉ－１］）

を計算し（ステップＳ１１）、ノートテーブルの所定の欄に書き込む（但し、ｉ＝１～ＮＡ，Ｔ［０］＝０）。すなわち、各ノートイベント間の相対時間が求まる。

【００３４】ここで、ＳＭＦにおいて、Δタイムは１拍の何分の１かを基本単位にした可変長符号で表され、値が小さいほど必要なバイト数は少なくて済む。例えば、値が１２７以下であれば１バイトでよいが、値が１２８

以上１６３８３以下であれば、２バイト必要となる。Δタイムの基本単位が細かいほど音楽的な表現力は高いと言えるが、それに従って必要なバイト数も増える。一方、実際に楽曲に使われているΔタイムを調べると、基本単位の１刻み（１ｔｉｃｋ）まで使っていないことが多く、したがって、ΔＴ［ｉ］の値が必要以上の容量を使って記録されていることが多い。

【００３５】そこで、実際に使用されている時間精度を求める為に、ノートテーブルに登録されている全ての相対時間ΔＴ［ｉ］に対する最大公約数ΔＴｓを算出する（ステップＳ１２）。この場合、最大公約数を求めるのが困難な場合は、適当な公約数を最大公約数ΔＴｓとする。次いで、ΔＴ［ｉ］をＳＭＦと同様の可変長符号で出力する（ステップＳ１３～Ｓ１７）。ただし、ΔＴ［ｉ］の値を可変長符号にする際に、ΔＴ［ｉ］をΔＴｓで除算した値ΔＴａ［ｉ］を符号化する（ステップＳ１５）。したがって、ノートΔ符号は図９のように最大公約数ΔＴｓと、ＮＡ個の値ΔＴａ［ｉ］（ｉ＝1～ＮＡ）により構成される。

【００３６】ここで、本演奏情報圧縮装置により圧縮された符号を伸長する際には、ΔＴａ［ｉ］の値を読み取りΔＴｓをかけ合わせることで元のΔＴ［ｉ］が復元され、したがって、ＳＭＦの持つ音楽的な表現力を失うことなくデータ量を削減することができる。例えばΔタイムの基本単位を一般的に良く使われる４８０分の１拍とし、ΔＴｓ＝１０である場合を例にすると、ＳＭＦでは１拍の長さΔＴ＝４８０や１／２拍の長さΔＴ＝２４０を表現するのに各々２バイト必要である。一方、本発明ではΔＴｓで除算して表現することによりΔＴ＝４８あるいはΔＴ＝２４を表現すればよいことになり、各々１バイトの使用で済む。また、１拍あるいは１／２拍に相当するΔタイムは使用頻度が高いので、これらが１つのΔタイムにつき１バイトずつ削減されれば、楽曲全体で相当の容量を削減することができる。

【００３７】コントローラΔ符号生成手段１４は、処理対象がノートテーブルではなくコントローラテーブルで

$$num[i]=f(num[i-1],num[i-2],\cdots,num[i-S])+\alpha[i]$$

但し、イベントの数をＮＡとして、

$i=(S+1),(S+2),\cdots,NA$

ここで、関数ｆ（）には種々のものが考えられるが、なるべく同じ値のα［ｉ］が繰り返して出現するようなものを選ぶことにより、２次符号生成手段４で効率の良い

$$num[i]=num[i-1]+\alpha[i]$$

但し、イベントの数をＮＡとして、

$i=2,3,\cdots,NA$

ここで、通常の楽曲においては、コード（和音）のルート音（根音）の平行移動量と同じ音程だけ移動したメロディーラインが存在することが多い。例えば「Ｃ」コードの小節で「ド、ド、ミ、ソ、ミ」というメロディーラインがあるという点以外はノートΔ符号生成手段１３と全く同じ処理であり、生成されるコントローラ符号の構成も符号の数がＮＡ個からＮＢ個に変わる以外は、図９に示すノートΔ符号と同じである。

【００３８】デュレーション符号生成手段１５はノートΔ符号生成手段１３とほぼ同じであり、図１０に従って処理を行う。まず、ノートテーブルに登録された各デュレーションの値の最大公約数Ｄｓを算出する（ステップＳ２１）。この場合、最大公約数を求めるのが困難な場合は、適当な公約数を最大公約数Ｄｓとする。デュレーション符号は図１１に示すように最大公約数Ｄｓと、ＮＡ個の値Ｄａ［ｉ］（ｉ＝1～ＮＡ）により構成され、最大公約数Ｄｓに続き、各デュレーションの値Ｄ［ｉ］をＤｓで割った値Ｄａ［ｉ］を可変長符号として出力する（ステップＳ２２～Ｓ２６）。ここで、デュレーションは前述したように、ＳＭＦのノートオンイベントとノートオフイベントの間のΔタイムに相当するので、ノートΔ符号生成手段１３において説明したのと同様な理由により、ＳＭＦに比べデータ量が削減される。

【００３９】ノートナンバ符号生成手段１６では、ノートテーブルに登録されているノートナンバに対し、以下の処理を施すことによりノートナンバ符号を生成する。ここで、あるノートナンバｎｕｍ［ｉ］を（１）式のようにそれ以前のＳ個のノートナンバ

ｎｕｍ［ｉ－１］、ｎｕｍ［ｉ－２］、…、ｎｕｍ［ｉ－Ｓ］

を変数とする関数ｆ（）と残差α［ｉ］で表す（ただし、ｎｕｍ［ｉ－１］はｎｕｍ［ｉ］の１つ前のノートナンバ、ｎｕｍ［ｉ－２］はｎｕｍ［ｉ］の２つ前のノートナンバを表す）。

【００４０】ノートナンバ符号は図１２に示すように、ｉ≦Ｓのイベントに対するノートナンバと、ｉ＞Ｓのイベントに対して残差α［ｉ］を時間順に並べたものにより構成される。したがって、圧縮時と伸長時で同じ関数ｆ（）を使えば、残差α［ｉ］からｎｕｍ［ｉ］を復元することができる。

$$num[i]=f(num[i-1],num[i-2],\cdots,num[i-S])+\alpha[i]\quad\cdots(1)$$

圧縮が可能となる。一例として（２）式のような関数を使った場合の効果を説明する。これはＳ＝１であり、１つ前のノートナンバとの差分をα［ｉ］とすることを意味する。ただしｉ＝１の場合はノートナンバそのものをノートナンバ符号として出力する。

$$num[i]=num[i-1]+\alpha[i]\quad\cdots(2)$$

インがある場合に、ルート音が２度高い「Ｄ」コードの小節において「レ、レ、#ファ、ラ、レ」というように、最初のメロディーラインを２度上げたメロディーラインが存在することが多い。

【００４１】この各々のメロディーラインをＳＭＦのノートナンバそのもので表すと、「６０，６０，６４，６

７，６０」、「６２，６２，６６，６９，６２」となり、この２つに共通のデータパターンは全くない。しかし前述のα［ｉ］で表現すると、どちらのメロディーラインもその２音目以降は「０，４，３，－７」となり同一のパターンとなる。このようにＳＭＦでは全く異なる２つのデータパターンを、本手法により同一のデータパターンに変換することができる。

【００４２】ＬＺ法では、同一のデータパターンが多いほど圧縮率が高くなるので、このようなノートナンバの表現方法により圧縮率が高くなることは明らかである。なお（１）式でＳ＝０とすると、

ｎｕｍ［ｉ］＝α［ｉ］

となり、ノートナンバそのものを符号化することになる。また関数ｆ()を何種類か用意しておき、最も適切な関数を選択して符号化するとともに、どの関数を使用したかという情報を合わせて符号化してもよい。

【００４３】ベロシティ符号生成手段１７もノートナンバ符号生成手段１６と同様である。ノートテーブルに登録されたある音符のベロシティｖｅｌ［ｉ］を（３）式のように、それ以前に出現したＴ個の音符のベロシティ

ｖｅｌ［ｉ－１］，ｖｅｌ［ｉ－２］，…，ｖｅｌ［ｉ－Ｔ］

を変数とする関数ｇ()と残差β［ｉ］で表す（ただし、ｖｅｌ［ｉ－１］はｖｅｌ［ｉ］の１つ前のベロシティ、ｖｅｌ［ｉ－２］はｖｅｌ［ｉ］の２つ前のベロシティ）。

ｖｅｌ［ｉ］＝ｇ（ｖｅｌ［ｉ－１］，ｖｅｌ［ｉ－２］，…，ｖｅｌ［ｉ－Ｔ］）＋β［ｉ］　（３）

但し、イベントの数をＮＡとして、

ｉ＝（Ｔ＋１），（Ｔ＋２），…，ＮＡ

次に、コントローラ符号生成手段１８について説明する。コントローラ符号は図１４に示すように、図６に示すコントローラテーブルに登録されたイベントの情報を時間順に並べたものである。各コントローラ符号は、イベントの種類を表すフラグＦとパラメータ（データバイト）で構成される。パラメータの個数はイベントの種類により異なる。イベントの種類は大きく分けて「通常イベント」と「連続イベント」の２つのタイプがある。フラグＦの最上位ビットが「１」、パラメータの最上位ビットが「０」に設定されているので、ＳＭＦと同様のランニングステータス表現（前のイベントと同じ種類のイベントの場合にフラグＦを省略すること）が可能になっている。

【００４５】ここで、ＳＭＦではイベントの種類をあらわすのに１バイトのＭＩＤＩステータスが使われている。一般的に使用される値は、８ｎ（ｈｅｘ）、９ｎ（ｈｅｘ）、Ａｎ（ｈｅｘ）、Ｂｎ（ｈｅｘ）、Ｃｎ（ｈｅｘ）、Ｄｎ（ｈｅｘ）、Ｅｎ（ｈｅｘ）、Ｆ０（ｈｅｘ）、ＦＦ（ｈｅｘ）のいずれかである（ただし、ｎ＝０～Ｆ（ｈｅｘ）、ｎはチャネル番号である）。「通常イベント」は、上記ＭＩＤＩステータスからノートオン８ｎ（ｈｅｘ）とノートオフ９ｎ（ｈｅｘ）を除いたものであるが、本発明では前述したようにチャネル番号を表現する必要が無いので、「通常イベント」のフラグの種類は７種類となる。従ってＭＩＤＩステータスに比べフラグＦは同じ値になる確率が高く、ＬＺ法を用いた場合の圧縮率が高まる。「通常イベント」の符号は、フラグＦの後に、ＳＭＦのＭＩＤＩステータス１バイトを除いたデータバイトを並べたものである。

【００４４】ベロシティ符号は図１３に示すように、ｉ≦Ｔのイベントに対するベロシティと、ｉ＞Ｔのイベントに対して残差β［ｉ］を時間順に並べたものにより構成される。したがって、圧縮時と伸長時で同じ関数ｇ()を使えば、残差β［ｉ］からｖｅｌ［ｉ］を復元でき、また、関数ｇ()を適当に選ぶことにより、同じデータパターンのβ［ｉ］が繰り返して出現することになり、ＬＺ法を用いた場合の圧縮率を改善することができる。

【００４６】また、ＳＭＦでは、特定の種類のイベントが一定数以上連続して出現し、各々のイベントのパラメータ値（データバイト）がほぼ一定の規則で変化する部分が存在することが多い。例えば「ピッチホイールチェンジ」イベントが使われている部分である。このイベントは、音符の音程を微妙に変えて音楽的な表現力を高める為のものであり、その性質上パラメータ値の異なる複数イベントが連続して使われることが多い。このようなイベントを「連続イベント」と呼び、このような部分を「連続イベントブロック」と呼ぶ。

【００４７】以下の説明では、「連続イベント」の一例として「ピッチホイールチェンジ」を取りあげるが、これに限定されるものではない。ＳＭＦの「連続イベントブロック」の一例を図１５に示す。この場合、各イベントのパラメータ値が異なる為、ＳＭＦの同一データパターンの長さは、（Δタイム・１バイト＋ステータス・１バイト）の計２バイトであり、この程度の長さではＬＺ法による圧縮の効果はほとんど得られない。

【００４８】そこで、コントローラテーブルの中で「ピッチホイールチェンジ」が一定数以上連続して出現し、パラメータ値がほぼ一定の規則で変化する領域に対し、以下の処理を施すことによりコントローラ符号を生成する。先ず、「ピッチホイールチェンジ」の数が一定数に満たないような場合は、前述した「通常イベント」として符号化する。そして、（４）式に示すように、連続イベントブロック内のイベントのパラメータｐ［ｉ］をそれ以前に出現したＵ個のイベントのパラメータ値

ｐ［ｉ－１］，ｐ［ｉ－２］，…，ｐ［ｉ－Ｕ］

を変数とする関数ｈ()と残差γ［ｉ］で表す（ただし、ｐ［ｉ－１］はｐ［ｉ］の１つ前のパラメータ値、ｐ［ｉ－２］はｐ［ｉ］の２つ前のイベントのパラメータ

値）。

【００４９】連続イベントの符号の構成は図１６に示すように、ピッチホイールチェンジが連続していることを意味するフラグＦに続き、１番目からＵ番目までのイベントに対してはパラメータの値そのものである。そして、（Ｕ＋１）番目以降のイベントでは、γ［ｉ］を時間順に並べたものである。

$$p[i]=h(p[i-1],p[i-2],\cdots,p[i-U])+\gamma[i] \quad (4)$$

ただし、連続イベントブロックのイベント数をNCとして

ｉ＝（Ｕ＋１），（Ｕ＋２），…，ＮＣ

関数ｈ()には種々のものが考えられるが、なるべく同じ値のγ［ｉ］が繰り返して出現するようなものを選ぶことにより、２次符号生成手段４において効率の良い圧縮が可能となる。一例として（５）式のような関数を使った場合の効果を説明する。これはＵ＝１であり、１つ前のパラメータとの差分をとることを意味する。

$$p[i]=p[i-1]+\gamma[i] \quad (5)$$

ただし、連続イベントブロックのイベント数をＮＣとして

ｉ＝ｉ＝２，３，…，ＮＣ

$$p[i]=e(p[i-1],p[i-2],\cdots,p[i-U],t[i],t[i-1],\cdots,t[i-U])+\gamma[i] \quad (6)$$

ただし、連続イベントブロックのイベント数をＮＣとして

ｉ＝（Ｕ＋１），（Ｕ＋２），…，ＮＣ

符号配置手段１９では、上記の各符号を図１８のような領域に配置して１次符号３を生成する。ヘッダは各符号の開始アドレスや長さといった管理情報と前述したチャネルマップを含んでいる。すでに説明したように各符号はＳＭＦに比べ、同一データの出現回数が多く、同一データパターンの長さも長いという性質を持っているが、さらに同一データパターンが近い距離で出現するように工夫をしている。まず、同じ種類の符号内で、同じデータ列が出現する確率が高いので、トラック順に同一種類の符号を配置している。また、ノートΔ符号とコントローラΔ符号とデュレーション符号は、全て時間情報であり、性質の異なるノートナンバ符号やベロシティ符号よりも同じデータ列が出現する確率が高いので、これらの距離が近くなるように配置している。

【００５０】次に、図１９で示した例に戻って、同一データパターンの長さがどの程度改善されるか具体的に検討する。ここで、各々のメロディは、５０個のノートオンイベントと５０個のノートオフイベントで構成されており、全てのΔタイムは１バイトであるとし、全てのイベントは３バイトであると仮定すると、各々のメロディのノートナンバは前述したように全て同じである。

【００５１】ＳＭＦにおいて、各々のメロディのデータ量は、

（１＋３）×５０×２＝４００バイト

である。各々のメロディのΔタイムとベロシティが全て同じならば、同一データパターンの長さは４００バイトになる。しかし両メロディ間の全てのΔタイムとノートオンのベロシティが異なっているとすると、ＳＭＦで同一データパターンの最大長は、ノートオフステータス、ノートナンバ、ベロシティの並びの３バイトである。この程度ではＬＺ法の圧縮はほとんど効果がない。

この方法によれば、図１５に示す領域は図１７のようなコントローラ符号に変換される。この場合、２番目以降のイベントのデータが全て同一の「１」になる為、ＬＺ法による圧縮率が高まる。また、コントローラ符号にはΔタイムが含まれないので、Δタイムがイベント毎に異なる場合でも、ＬＺ法における圧縮率低下の影響が少ない。また場合によっては、（４）式の代わりに、イベントの時間情報も変数に使った（６）式のような関数ｅ()を使っても良い。ただし、ｔ［ｉ］はパラメータを求めるイベントの時間、ｔ［ｉ－１］はその１つ前のイベントの時間である。

【００５２】一方、本発明では、Δタイム、ノートナンバ、ベロシティを分離して符号化しているので、少なくともノートナンバ符号の中で５０バイトの長さの同一データパターンが出現する。また前述したようにＳＭＦのベロシティが全く異なる場合でも、ベロシティ符号の中では同一データパターンが出現することが多い。従ってＬＺ法による圧縮率は明らかに改善される。以上の説明から分かるように１次符号３は、ＳＭＦの持つ音楽的な情報量を全く落とすことなくデータ量が削減されていると同時に、ＳＭＦに比べ同一のデータパターンの長さが長く、出現回数が多く、しかもそれらが近い距離で出現するようの性質を持っているので、２次符号生成手段４において効果的な圧縮を行うことができる。また、この１次符号３そのものも、かなり圧縮されたデータ量となっているので、この１次符号３を直接出力するようにしてもよい。

【００５３】２次符号生成手段４においては、１次符号生成手段２の出力３に対して、ＬＺ法による圧縮を行う。ＬＺ法は、ｇｚｉｐ、ＬＨＡといった圧縮プログラムで広く使われている手法である。これは入力データの中から、同一のデータパターンを捜し、もし存在すれば、（過去に出現したパターンへの距離、パターンの長さ）という情報に置き換えて表現することでデータ量を削減する。例えば、

”ＡＢＣＤＥＡＢＣＤＥＦ”

というデータを、”ＡＢＣＤＥ”が繰り返しであるの

で、
”ＡＢＣＤＥ（５，５）Ｆ”
という情報に置き換える。なお、圧縮符号（５，５）は５文字戻って５文字コピーすることを表す。

【００５４】処理の概要は次のようになる。２次符号５の生成は、処理位置を１次符号３の先頭から順次移動させて行う。処理位置のデータパターンが、それ以前の一定の範囲内のデータパターンと一致する場合は、処理位置からそのデータパターンまでの距離と、一致したデータパターンの長さを２次符号５として出力し、処理位置を２つ目のデータパターンの終わりに移動させ、処理を続ける。処理位置のデータパターンが、それ以前の一定の範囲内のデータパターンと一致しなければ、１次符号３をコピーして２次符号５として出力する。

【００５５】以上の説明から明らかなように、２つの同一のデータ領域が大きいほど、圧縮率は高くなる。また同一のデータ領域の距離は一定範囲内である必要がある。前述したように楽曲の中では、同じようなメロディーが繰り返し使われるが、ＳＭＦのままでそれらのデータを比較すると、完全に同一のデータ列の繰り返しであることは少なく、むしろノートナンバは同じであるがベロシティは異なるといったように、どこか一部分異なっていることが多い。

【００５６】一方，本発明では、性質の同じデータを独立した領域にまとめると同時に、各領域で同一のデータがなるべく多く出現するような処理を行ない、さらに性質の近い領域どうしをなるべく近くに配置することにより、ＬＺ法の圧縮率が高まるので、最終的な２次符号５は十分容量の小さなものになる。なお、以上詳述したフォーマット並びに処理手順は一例であり、その主旨を逸脱しない範囲において種々の変更を加えることができる。また、演奏情報としてＳＭＦを例に取ったが、ＳＭＦに限らずこれに類似の演奏情報に対して本発明を適用してデータ容量を効率よく削減することができる。

【００５７】次に、図２０～図２８を参照して上記の１次符号３又は２次符号５を復号するための演奏情報復号装置について説明する。図２０は演奏情報復号装置を示すブロック図，図２１は図２０の２次符号復号手段の処理を説明するためのフローチャート、図２２は図２０の１次符号復号手段の処理を説明するためのフローチャート、図２３は図２２のトラック復号処理を詳しく説明するためのフローチャート、図２４は図２３のノートイベント復号処理を詳しく説明するためのフローチャート、図２５は図２４のノートイベント復号処理により復元されたノートオンイベントを示す説明図、図２６は図２４のノートイベント復号処理により復元されたノートオフキューを示す説明図、図２７は図２３のコントロールイベント復号処理を詳しく説明するためのフローチャート、図２８は図２７の処理により復元されたコントロールイベントを示す説明図である。

【００５８】図２０では圧縮処理とは逆に、ＬＺ法で圧縮された入力データ２１が２次符号復号手段２３により音程と、音の強さと、音の長さとその他の情報に分離された１次符号３に復号され、次いで１次符号復号手段２４により元の音符（出力データ２５）に復元される。制御手段２６はスイッチ２２により、入力データ２１が図１に示す２次符号５である場合に２次符号復号処理に続き１次符号復号処理を行うように制御し、入力データ２１が図１に示す１次符号３である場合に１次符号復号処理のみを行うように制御する。

【００５９】ここで、２次符号５であるか又は１次符号３であるかの判定は、キーボード、マウス、ディスプレイ等の図示しない入出力装置を使用してオペレータが指定してもよいし、圧縮された情報に対して符号化方法の種類を示す情報を符号化時に付加し、復号時にこの情報を自動的に判別するようにしてもよい。

【００６０】次に、図２１を参照して２次符号復号手段２３の復号処理を説明する。入力データ１１（２次符号５）を先頭から読み込み（ステップＳ１０１）、次いで圧縮データの部分であるかすなわちＡＢＣＤＥ（５，５）の「ＡＢＣＤＥ」の部分（＝非圧縮データ）であるか又は「（５，５）」の部分（＝圧縮データ）であるかを判定する（ステップＳ１０２）。

【００６１】そして、圧縮データの部分である場合には過去に出現した同一パターンを参照してそれをコピーして出力し（ステップＳ１０３）、他方、非圧縮データの部分である場合にはそのまま出力する（ステップＳ１０４）。以下、入力データ１１を全て復号するまでこの処理を繰り返すと（ステップＳ１０５→Ｓ１０１）、図１８に示すような配置の１次符号３が復元される。

【００６２】次に、図２２及び図１８に示す１次符号３を参照して１次符号復号手段２４の復号処理を説明する。先ず、１次符号３のヘッダを読み込む（ステップＳ１１１）。ヘッダには総トラック数Ｎ、ノートΔ符号からコントロール符号までの各符号領域の先頭アドレス、チャネルマップ、時間分解能等の情報が符号化の際に記録されているので、このヘッダ情報に基づいてＳＭＦのヘッダを作成して出力する（ステップＳ１１２）。

【００６３】次にトラック番号ｉを「１」にセットし（ステップＳ１１３）、図２３に詳しく示すトラック復号処理を行う（ステップＳ１１４）。次いでトラック番号ｉが総トラック数Ｎより小さいか否かをチェックし（ステップＳ１１５）、もし小さければトラック番号ｉを１つインクリメントし（ステップＳ１１６）、ステップＳ１１４に戻ってトラック復号処理を繰り返す。そして、ステップＳ１１５においてトラック番号ｉが総トラック数Ｎより小さくない場合にこの１次符号復号処理を終了する。

【００６４】図２３に詳しく示すトラック復号処理では、先ず、処理で使用する変数を初期化する（ステップ

Ｓ１２１）。具体的は、処理中のノートイベントの番号を示す変数ｊを「１」にセットし、処理中のコントローライベントの番号を示す変数ｋを「１」にセットし、ノート終了フラグとコントローラ終了フラグをクリアする。ここで、ノート終了フラグは処理トラックの全てのノートイベントの復号が終了したことを示し、コントローラ終了フラグは処理トラックの全てのコントローライベントの復号が終了したことを示す。

【００６５】次に処理トラック番号ｉのノートΔ符号の最大公約数ΔＴｓｎと、コントローラΔ符号の最大公約数ΔＴｓｃとデュレーション符号の最大公約数Ｄｓを読み出す（ステップＳ１２２）。そして、ｊ番目のノートΔ符号ΔＴａｎ［ｊ］とｋ番目のコントローラΔ符号ΔＴａｃ［ｋ］を読み出し、（７）式のように各々最大公約数ΔＴｓｎ、ΔＴｓｃを乗じてΔＴｎ［ｊ］、ΔＴｃ［ｋ］を算出する（ステップＳ１２３）。

$$\Delta Tn[j] = \Delta Tan[j] \times \Delta Tsn$$
$$\Delta Tc[k] = \Delta Tac[j] \times \Delta Tsc \quad (7)$$

さらに、（８）式のようにトラックの先頭を基準とした時刻Ｔｎ［ｊ］、Ｔｃ［ｋ］に変換する（ステップＳ１２４）。

$$Tn[j] = Tn[j-1] + \Delta Tn[j]$$
$$Tc[k] = Tc[k-1] + \Delta Tc[k]$$
$$\text{ただし、} Tn[0] = Tc[0] = 0 \quad (8)$$

なお、ステップＳ１２３、Ｓ１２４では、ノート終了フラグがセットされている場合にはΔＴｎ［ｊ］、Ｔｎ［ｊ］の算出は行わず、また、コントローラ終了フラグがセットされている場合にはΔＴｃ［ｋ］，Ｔｃ［ｋ］の算出は行わない。

【００６６】次に、出力すべきノートオフイベントの有無をチェックし（ステップＳ１２５）、出力すべきデータが有る場合にはＳＭＦとしてノートオフイベントを出力する（ステップＳ１２６）。なお、ステップＳ１２５、Ｓ１２６については後述（図２４のステップＳ１４４）する。次に、復号処理の選択を行う。先ず、コント

$$num[j] = f(num[j-1], num[j-2], \cdots, num[j-S]) + \alpha[j] \quad (j > S)$$
$$num[j] = \alpha[j] \quad (j \leq S)$$

ただし、Ｓは関数ｆ（）の変数の個数　(9)

同様に、ｊ番目のベロシティ符号β［ｊ］を読み取り、圧縮処理において使用した関数ｇ（）を用いて（１０）

$$vel[j] = g(vel[j-1], vel[j-2], \cdots, vel[j-T]) + \beta[j] \quad (j > T)$$
$$vel[j] = \beta[j] \quad (j \leq T)$$

ただし、Ｔは関数ｇ()の変数の個数　(10)

次いで、Ｔｎ[ｊ]、ｎｕｍ［ｊ］，ｖｅｌ［ｊ］を用いて図２５に示すようなノートオンイベントを出力する（ステップＳ１４３）。なお、ＳＭＴのΔタイムΔＴは、Ｔｎ［ｊ］の直前に出力したイベントの時刻Ｔｂを使って式（１１）に従って求め、出力する。

$$\Delta T = Tn[j] - Tb \quad (11)$$

図２５に示すノートオンイベントにおけるステータスバイトの上位４ビットはノートオン「９（ｈｅｘ）」を表

ローラ終了フラグをチェックし（ステップＳ１２７）、セットされている場合には図２４に詳しく示すノートイベント復号処理を行う（ステップＳ１２８）。

【００６７】コントローラ終了フラグがセットされていない場合にはノート終了フラグをチェックし（ステップＳ１２９）、セットされている場合には図２７に詳しく示すコントローライベント復号処理を行う（ステップＳ１３０）。２つのフラグが共にセットされていない場合にはＴｎ［ｊ］とＴｃ［ｋ］を比較し（ステップＳ１３１）、Ｔｎ［ｊ］が小さければノートイベント復号処理（ステップＳ１２８）を、そうでなければコントローライベント復号処理（ステップＳ１３０）を行う。

【００６８】ノートイベント復号処理の後には、処理トラックＮの全てのノートイベントを処理したか否かをチェックし（ステップＳ１３２）、処理が終了している場合にはノート終了フラグをセットし（ステップＳ１３３）、ステップＳ１３８に進み、そうでなければ変数ｊを１つインクリメントし（ステップＳ１３４）、ステップＳ１２３に戻る。また、コントローライベント復号処理の後には、処理トラックＮの全てのコントローライベントを処理したか否かをチェックし（ステップＳ１３５）、処理が終了している場合にはコントローラ終了フラグをセットし（ステップＳ１３６）ステップＳ１３８に進み、そうでなければ変数ｋを１つインクリメントし（ステップＳ１３７）、ステップＳ１２３に戻る。

【００６９】ステップＳ１３８ではノート終了フラグとコントローラ終了フラグの両方がセットされているか否かをチェックし、両方がセットされている場合にはこのトラック復号処理を終了し、そうでなければステップＳ１２３に戻ってこのトラック復号処理を繰り返す。

【００７０】図２４に詳しく示すノートイベント復号処理では、先ず、ｊ番目のノートナンバ符号α［ｊ］を読み取り、圧縮処理において使用した関数ｆ（）を用いて（９）式に従ってノートナンバｎｕｍ［ｊ］を算出する（ステップＳ１４１）。

式に従ってベロシティｖｅｌ［ｊ］を算出する（ステップＳ１４２）。

し、下位４ビットはチャネルマップから得られる番号が続く。このステータスバイトの後にはノートナンバとベロシティの各バイトが続く。

【００７１】次にノートオフイベントの登録を行う（ステップＳ１４４）。具体的にはデュレーション符号Ｄａ［ｊ］を読み取って、（１２）式に従いノートオフイベントの時刻Ｔoff を算出し、この時刻Ｔｏｆｆとノートナンバｎｕｍ［ｊ］を図２６に示すようなノートオフキューに登録する。このノートオフキューでは、使用されているエントリの数を保持するとともに、ノートオフ時刻Ｔｏｆｆが先頭から小さい順に並ぶように管理される。

$$Toff[j]=Da[j]\times Ds+Tn[j] \quad (12)$$

前述した図２３のステップＳ１２５においては、Ｔｎ［ｊ］とＴｃ［ｋ］の内の値が小さいほうＴｍをノートオフキューの先頭のＴｏｆｆ［ｎ］（ｎ＝１～エントリ総数Ｎ）から順に比較する。Ｔｏｆｆ［ｎ］＜ＴｍであるエントリがあればステップＳ１２６に進み、ノートオフイベントを出力する。ステップＳ１２６では、前述したノートオフイベントをＳＭＦとして出力する。

【００７２】次に図２７を参照してコントローライベント復号処理を詳しく説明する。この処理では図２８に示すようにΔタイム、ステータス及びパラメータより成るコントローライベントが復元され、先ず、Ｔｃ［ｋ］と直前に出力したイベントの時刻Ｔｂを使って（１３）式に従ってＳＭＴのΔタイムΔＴを求め、出力する（ステップＳ１５１）。

$$\Delta T=Tc[k]-Tb \quad (13)$$

次にコントローラ符号領域からイベントの種類を表すイベントフラグＦ［ｋ］を読み取り、Ｆ［ｋ］が「通常イベント」であるか、「連続イベント」であるか又は「ランニングステータス」であるかを判定する（ステップＳ１５２）。ここで、連続イベントブロック内では、図示省略１６に示すように、２番目以降のイベントはイベントフラグが省略された「ランニングステータス」状態で記録されている。

【００７３】Ｆ［ｋ］が「通常イベント」である場合には、処理イベントの連続イベントブロック内における順番を示す変数ｍを「０」にリセットし（ステップＳ１５３）、次いでチャネルマップを参照してＳＭＦのステータスバイトを作成して出力する（ステップＳ１５４）。さらにイベントの種類に応じて必要なバイト数をコントローラ符号領域から読み出し、この読み出した値がＳＭＦのパラメータ（データバイト）であるのでこれを出力する（ステップＳ１５５）。

【００７４】Ｆ［ｋ］が「連続イベント」である場合には、連続イベントブロック内における順番を示す変数ｍを「１」にセットし（ステップＳ１５６）、次いでチャネルマップを参照してＳＭＦのステータスバイトを作成して出力する（ステップＳ１５７）。なお、ｍ≧２の場合のステータスバイトはｍが「１」の場合のステータスバイトを利用する。そして、この「連続イベント」の場合には、パラメータ符号γ［ｍ］を読み出し、圧縮処理と同じ関数ｈ（）を使い、（１４）式に従ってＳＭＦのパラメータｐ［ｍ］を作成し、出力する（ステップＳ１５８）。

$$p[m]=h(p[m-1],p[m-2],\cdots,p[m-U])+\gamma[m] \quad (m>U)$$

$$p[m]=\gamma[m] \quad (m\leqq U) \quad (14)$$

ただし、Ｕは関数ｈ（）の変数の個数

Ｆ［ｋ］が「ランニングステータス」である場合には、変数ｍの値をチェックし（ステップＳ１５９）、ｍが「０」より大きければｍを１つインクリメントし（ステップＳ１６０）、「連続イベント」側のステップＳ１５７に進む。他方、ｍが「０」であれば「通常イベント」側のステップＳ１５４に進む。

【００７５】次に本発明の、ＳＭＦを時間軸上で２つ以上のブロックに分割し、少なくとも最初のブロックは前記１次圧縮及び／又は前記２次圧縮を行わないことを特徴とする演奏情報圧縮装置について説明する。

【００７６】図２９に本発明による圧縮を行ったときの第１の実施例を示す。従来の方法では、元の演奏情報はそのまま１曲ごとに圧縮を行っているが、演奏情報を着信メロディなどで使用する場合には、電話受信後、直ちに着信メロディが再生されなければならず、圧縮されたファイルを解凍するために端末側の処理速度を高速化する必要がある。そこで、図２９のように、演奏情報のファイルを時系列上にブロック１とブロック２の２つのブロックに分割し、それぞれのブロック単位で圧縮を行う。このとき、ブロック１はできるだけ端末側で即座にデータを解凍できるように１次符号化のみの圧縮を施し、２次符号化は行わない。ブロック２は全体のメモリー容量や配信ファイルサイズを小さくするために、通常の１次符号化ならびに２次符号化による圧縮を行う。このようにして、１次符号化のみ施された圧縮ブロック１と１次符号化及び２次符号化が施された圧縮ブロック２とを一つのファイルとして配信伝送し、端末側のメモリに記録させる。

【００７７】ブロック１は、例えば最初の５秒分など、ある程度短めにしておき、電話に対して着信があったとき、まず、圧縮されたブロック１から解凍し、ブロック１に該当する部分の着信メロディを再生しながら、同時に圧縮されたブロック２を解凍する。そのため、ブロック１はブロック２の解凍を行うための十分な時間があれ

ば、可能な限り短い方が良い。これは圧縮されたブロック１を解凍する作業時間を短縮する点においても重要である。

【００７８】図３５に第１実施例のエンコードのフローを示す。まず、ＳＭＦを取りこみ（ステップ４０１）、ステップ４０２にてＳＭＦを時間軸上で２分割し、最初の５秒分をブロック１、残りの部分をブロック２とする。そして、ステップ４０３でブロック１を１次符号化、これによりステップ４０４にてブロック１を圧縮した圧縮ブロック１を生成する。そして、ステップ４０５にてブロック２を１次符号化及び２次符号化し、ステップ４０６にて圧縮ブロック２を生成する。更に、圧縮ブロック１及び圧縮ブロック２を一つにまとめて（ステップ４０７）、エンコードを終了する。

【００７９】次に再生時の動作を図３１及び図３２を参照しながら説明する。電話の着信が発生したとき、楽曲取り込み手段１３４により、発信相手に対応する楽曲ファイルを情報記憶手段１３５に取り込む。既に情報記憶手段１３５に楽曲ファイルが収録されている場合には、直接ここから解凍を行ってもよい。まず圧縮ブロック１の解凍を始める。この際、圧縮ブロック１は１次符号化のみ施されているので、図３１の１次符号復号手段１３１によって復号を行い、直ちに演奏情報の再生を行う。そして演奏情報の再生を行いながら、圧縮ブロック２の解凍を始める。圧縮ブロック２は図３１の２次符号復号手段１３２によって、２次復号化が行われ、次に１次符号復号手段１３１によって１次復号化が行われる。ブロック１のファイルサイズは、この２次復号化が終了し、ブロック２の演奏情報が再生可能となるタイミングまでに時間的な猶予が持てる程度の大きさが好ましい。また、演奏情報はテンポをベースに記述されている場合があるので、再生テンポが端末側で最大（最も早い）となるように設定されている場合にも間に合うようにしなければならない。展開した楽曲情報は情報記憶手段１３５に記録される。そして楽曲再生手段１３６により、情報記憶手段１３５に展開された演奏情報が再生される。なお、上述した実施例では発信相手に対応する楽曲ファイルを選んでいるが、発信相手の全てに同じ着信メロディを使用し、電話の使用者の好みに応じて楽曲を切り替えるようにしても良い。

【００８０】次に、本発明による圧縮を行ったときの第２の実施例について図３０を参照して説明する。これは、第１の実施例よりも端末の処理が遅い場合に、全体の圧縮率はやや下がるが、ブロック１の部分については元のデータをそのまま用いるものである。図３６に第２実施例のエンコードのフローを示す。なお、ブロック１の１次符号化を行わない点を除いて上述した第１の実施例と同じであるため説明を省略する。また、再生時の動作についても図３１に示す第１の実施例と同様の端末構成で、図３３に示すように圧縮ブロック１の解凍を行わない点が異なるものであり説明は省略する。

【００８１】次に、本発明による圧縮を行ったときの第３の実施例について図３４を参照して説明する。これは、ある程度端末側の処理速度の高速化が見込まれる場合に、ブロック１にも１次符号化ならびに２次符号化を施すものである。図３７に第３実施例のエンコードのフローを示すがこれについてもブロック１に対して１次符号化及び２次符号化を施す以外には上述した実施例と同様であるため、説明を省略する。次に再生時のフローについてであるも上述した実施例と同様であり、ブロック１の部分のみがそれぞれの圧縮内容による違いが生ずるのみであるので、説明は省略する。

【００８２】なお、上記は着信メロディについて説明を行ったが、カラオケやＢＧＭであっても、再生指定時に直ちに演奏を開始することができる点で、実用に耐えうるものとなるので、これらのような使用形態に適用することも可能である。

【００８３】

【発明の効果】以上説明したように本発明によれば、ＬＺ法により圧縮する前に予め、同一のデータパターンの長さが長く、出現回数が多く且つ近い距離で出現するように、演奏情報を音程と、強さと、長さとその他の情報に分離し、各情報をそれぞれ独立した領域に配置した１次符号を生成し、この１次符号をＬＺ法により圧縮するので、演奏情報のデータ量を効率的に圧縮することができるが、このような装置に於いて、ＳＭＦを２つ以上のブロックに分割し、少なくとも最初のブロックは前記１次圧縮及び/または前記２次圧縮を行わないことを特徴とする装置を提供することで、圧縮した状態からの再生をすぐに行える端末を提供する事ができる。

【００８４】また、１次符号として演奏情報を音符の音程領域と、音符の強さ領域と、音符の長さ領域とその他の領域の少なくとも４つの領域に分離して符号化するので、元の演奏情報のもつ演奏品位を全く失うことなく、従来に比べ大幅にデータ容量が削減でき、したがって、データを保存するのに小容量の記録媒体を用いることができコストを削減することができ、また、通信回線を介してデータを伝送する場合にもコストを削減することができるとともに、伝送時間を削減することができる。また、大量の演奏情報を扱う通信カラオケや着信メロディなどの音楽データベースや端末のメモリ容量節約特に効果が大きい。

【図面の簡単な説明】

【図１】本発明に係る演奏情報圧縮装置の一例を示すブロック図である。

【図２】図１の１次符号生成手段の一例を詳細に示すブロック図である。

【図３】図２のチャネル分離手段により作成されるチャネルマップを示す説明図である。

【図４】図２の解析手段の処理を説明するためのフロー

チャートである。

【図５】図２の解析手段により作成されるノートテーブルを示す説明図である。

【図６】図２の解析手段により作成されるコントローラテーブルを示す説明図である。

【図７】音符を表現するＳＭＦのΔタイムと本実施例のデュレーションの関係を示す説明図である。

【図８】図２のノートΔ符号生成手段の処理を説明するためのフローチャートである。

【図９】図２のノートΔ符号生成手段により生成されるノートΔ符号を示す説明図である。

【図１０】図２のデュレーション符号生成手段の処理を説明するためのフローチャートである。

【図１１】図２のデュレーション符号生成手段により生成されるデュレーション符号を示す説明図である。

【図１２】図２のノートナンバ符号生成手段により生成されるノートナンバ符号を示す説明図である。

【図１３】図２のベロシティ符号生成手段により生成されるベロシティ符号を示す説明図である。

【図１４】図２のコントローラ符号生成手段により生成されるコントローラ符号を示す説明図である。

【図１５】ＳＭＦの連続イベントブロックを示す説明図である。

【図１６】本実施例の連続イベントブロックを示す説明図である。

【図１７】図１６の連続イベントブロックの効果を示す説明図である。

【図１８】図２の符号配置手段により並べ替えられた１次符号を示す説明図である。

【図１９】ＳＭＦのフォーマットを示す説明図である。

【図２０】演奏情報復号装置を示すブロック図である。

【図２１】図２０の２次符号復号手段の処理を説明するためのフローチャートである。

【図２２】図２０の１次符号復号手段の処理を説明するためのフローチャートである。

【図２３】図２２のトラック復号処理を詳しく説明するためのフローチャートである。

【図２４】図２３のノートイベント復号処理を詳しく説明するためのフローチャートである。

【図２５】図２４のノートイベント復号処理により復元されたノートオンイベントを示す説明図である。

【図２６】図２４のノートイベント復号処理により復元されたノートオフキューを示す説明図である。

【図２７】図２３のコントローライベント復号処理を詳しく説明するためのフローチャートである。

【図２８】図２７の処理により復元されたコントローライベントを示す説明図である。

【図２９】本発明に係る演奏情報圧縮装置による圧縮方法の第１の実施例を示す図である。

【図３０】本発明に係る演奏情報圧縮装置による圧縮方法の第２の実施例を示す図である。

【図３１】本発明に係る演奏情報圧縮装置の構成を示す図である。

【図３２】第１の実施例により圧縮された演奏情報の解凍方法を示す図である。

【図３３】第２の実施例により圧縮された演奏情報の解凍方法を示す図である。

【図３４】本発明に係る演奏情報圧縮装置による圧縮方法の第３の実施例を示す図である。

【図３５】本発明に係る演奏情報圧縮装置による圧縮方法の第１の実施例の動作を示す図である。

【図３６】本発明に係る演奏情報圧縮装置による圧縮方法の第２の実施例の動作を示す図である。

【図３７】本発明に係る演奏情報圧縮装置による圧縮方法の第３の実施例の動作を示す図である。

【符号の説明】

１　入力データ
２　１次符号生成手段
３　１次符号
４　２次符号生成手段
５　２次符号
１１　チャネル分離手段
１２　解析手段
１３　ノートΔ符号生成手段
１４　コントローラΔ符号生成手段
１５　デュレーション符号生成手段
１６　ノートナンバ符号生成手段
１７　ベロシティ符号生成手段
１８　コントローラ符号生成手段
１９　符号配置手段
２１　入力データ
２２　スイッチ
２３　２次符号復号手段
２４　１次符号復号手段
２５　出力データ
２６　制御手段

【図２５】

ノートオンイベント

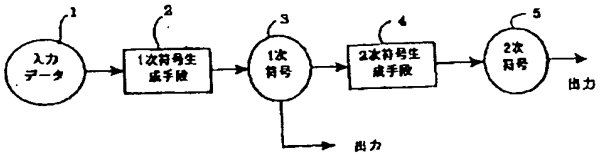
Δタイム	ステータス (9x hex)	ノートナンバ	ベロシティ

【図２８】

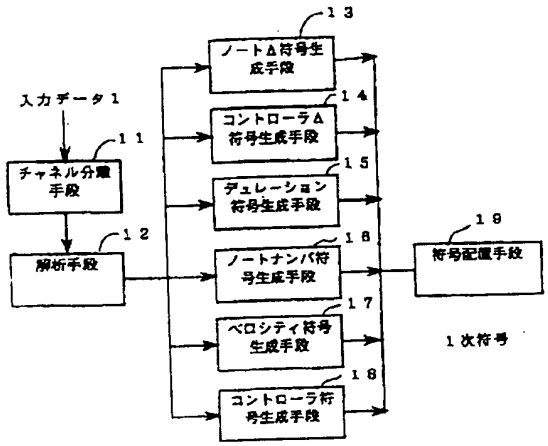
コントローライベント

Δタイム	ステータス	パラメータ

【図１】



【図２】



【図３】

チャネルマップ

トラック	チャネル番号
1	2
2	3
3	5
4	1

【図５】

ノートテーブル

	時間	ノートナンバ	ベロシティ	デュレーション	ノートオフ番号	ΔT
ノート1	480	60	80	240	1	
ノート2	840	62	80		0	
ノートNA						

【図６】

コントローラテーブル

	時間	データ
イベント1		
イベント2		
イベントNB		

【図７】

SMF

Δタイム（可変長）	
ノートオンステータス(1バイト)	ノートオンイベント
ノートナンバ(1バイト)	
ベロシティ(1バイト)	
Δタイム（可変長）	デュレーションに相当
ノートオフステータス(1バイト)	ノートオフイベント
ノートナンバ(1バイト)	
ベロシティ(1バイト)	

【図９】

ノートΔ符号

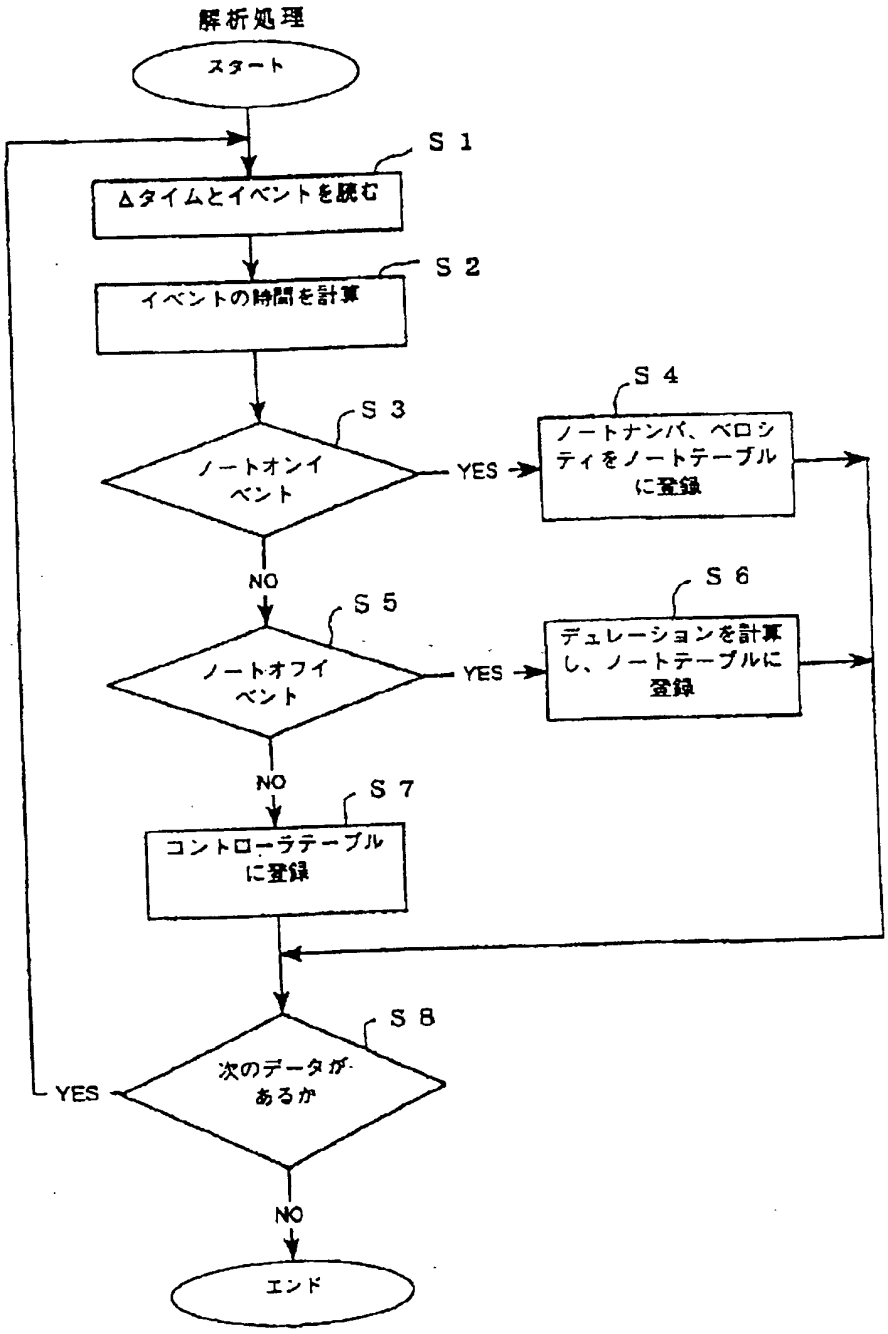
ΔTs
ΔTe[1]
ΔTe[2]
ΔTe[NA]

【図１１】

デュレーション符号

Ds
De[1]
De[2]
De[NA]

【図４】



【図１２】

ノートナンバ符号

num[1]	ノートナンバ
num[2]	
num[S]	
α[S+1]	残差
α[S+2]	
α[NA]	

【図１３】

ベロシティ符号

vel[1]	ベロシティ
vel[2]	
vel[T]	
β[T+1]	残差
β[T+2]	
β[NA]	

【図１７】

コントローラ符号

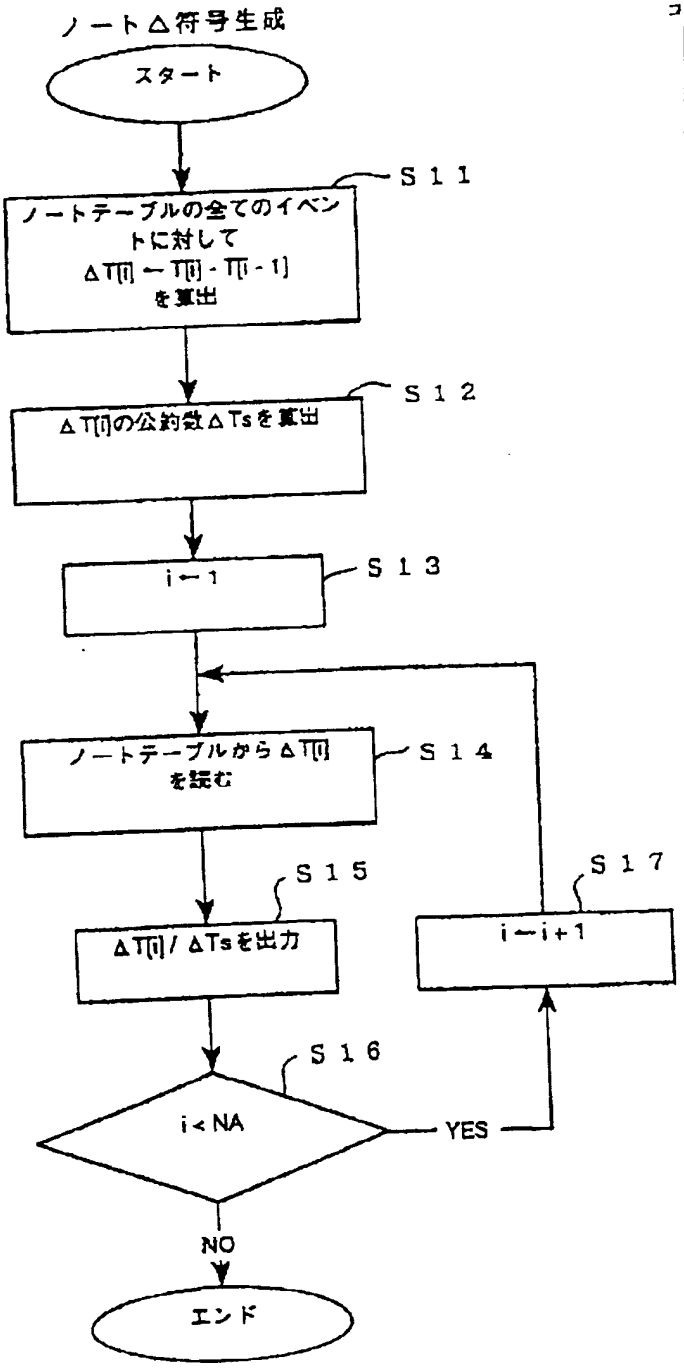
フラグF
初期値 "8192"
残差 "1"
残差 "1"
残差 "1"
残差 "1"
残差 "1"
残差 "1"

【図２６】

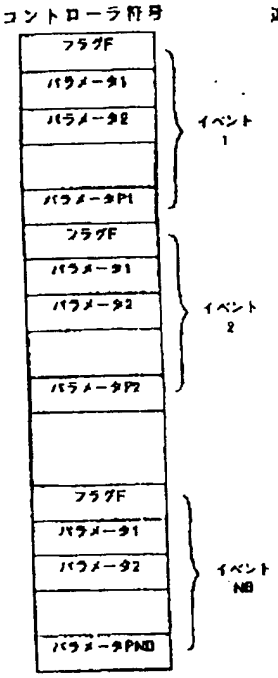
ノートオフキュー

	Toff	ノートナンバ
エントリ1	1000	64
エントリ2	1100	64

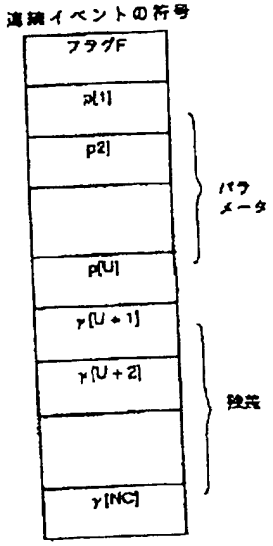
【図８】



【図１４】



【図１６】

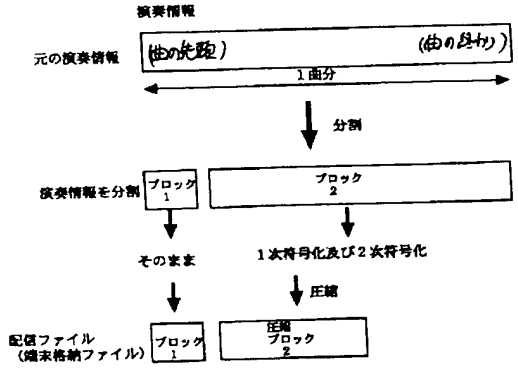


【図１５】

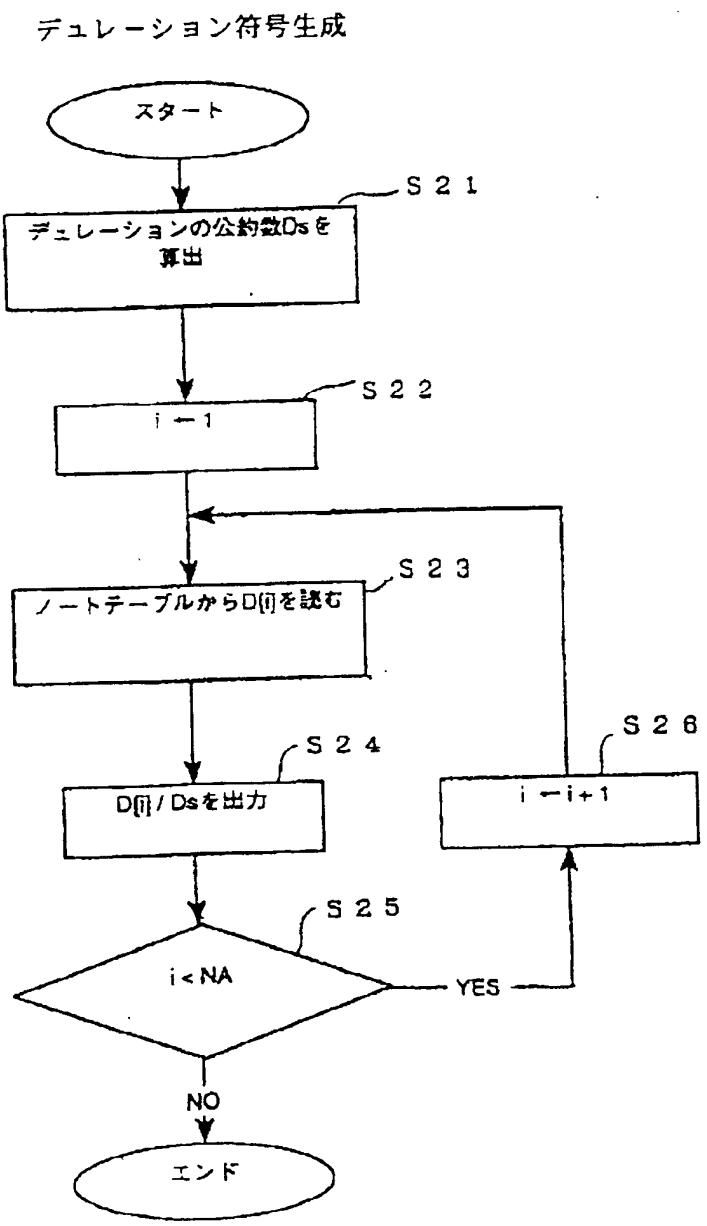
連続イベントブロック

Δタイム	ステータス	パラメータ
10	224(ピッチホイールチェンジ)	8192
20	224(ピッチホイールチェンジ)	8193
20	224(ピッチホイールチェンジ)	8194
10	224(ピッチホイールチェンジ)	8195
30	224(ピッチホイールチェンジ)	8196
10	224(ピッチホイールチェンジ)	8197
20	224(ピッチホイールチェンジ)	8198

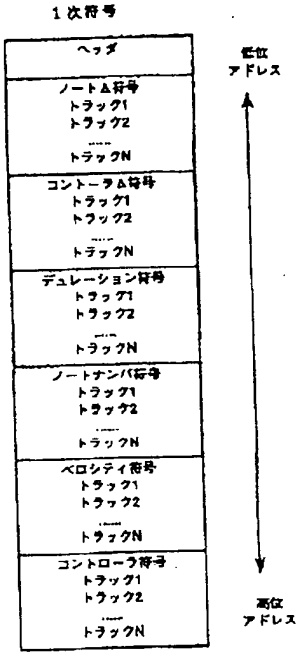
【図３０】



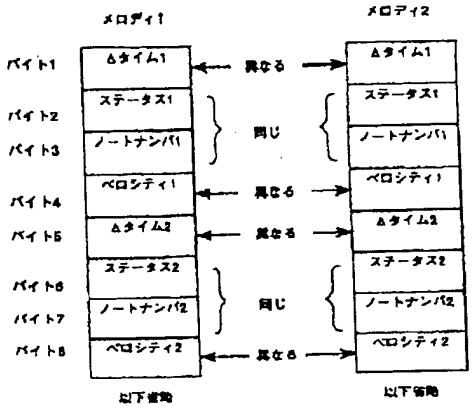
【図１０】



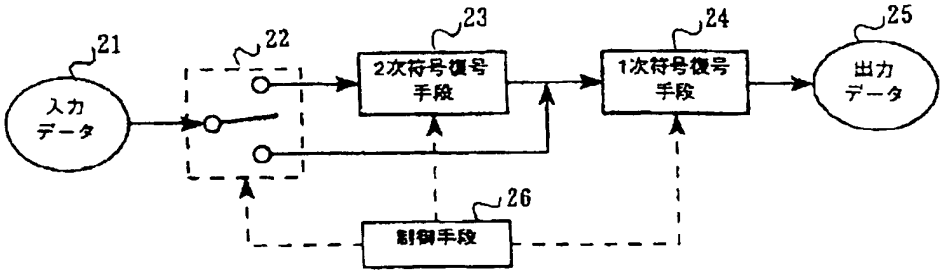
【図１８】



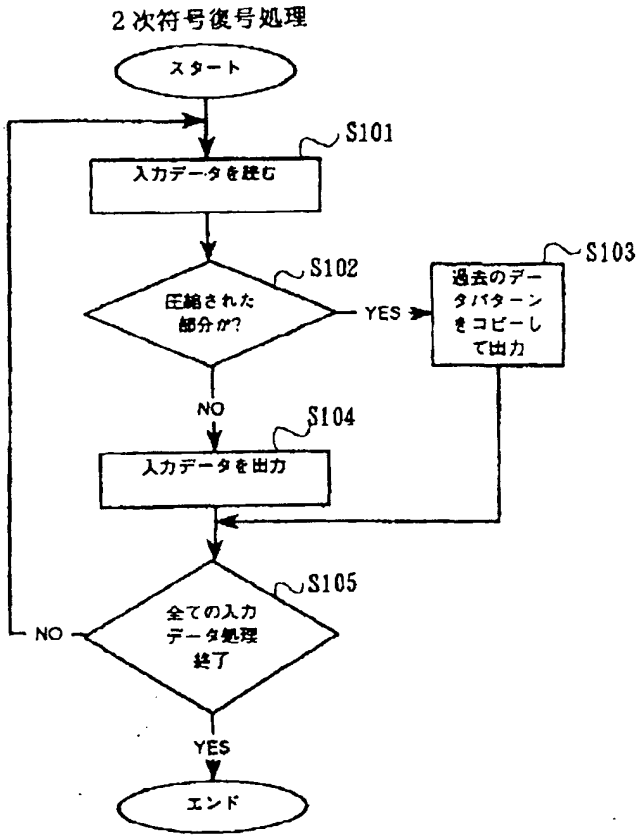
【図１９】



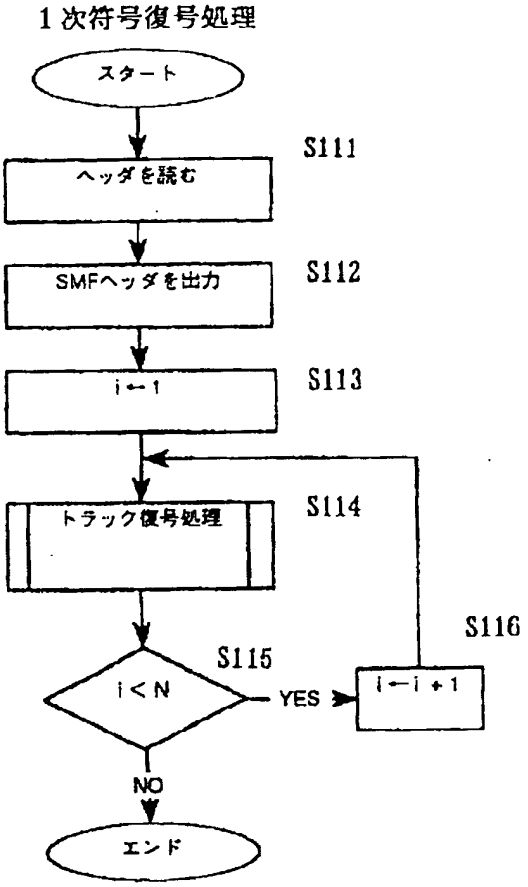
【図２０】



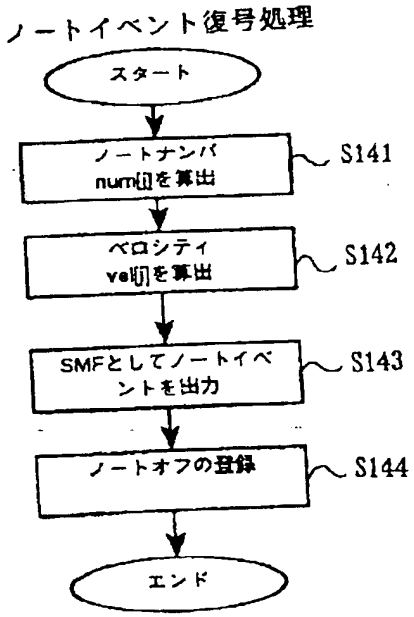
【図２１】



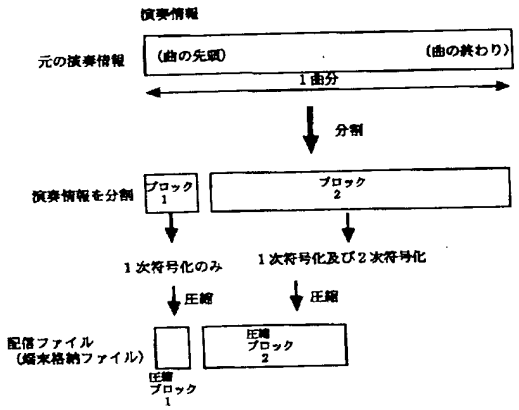
【図２２】



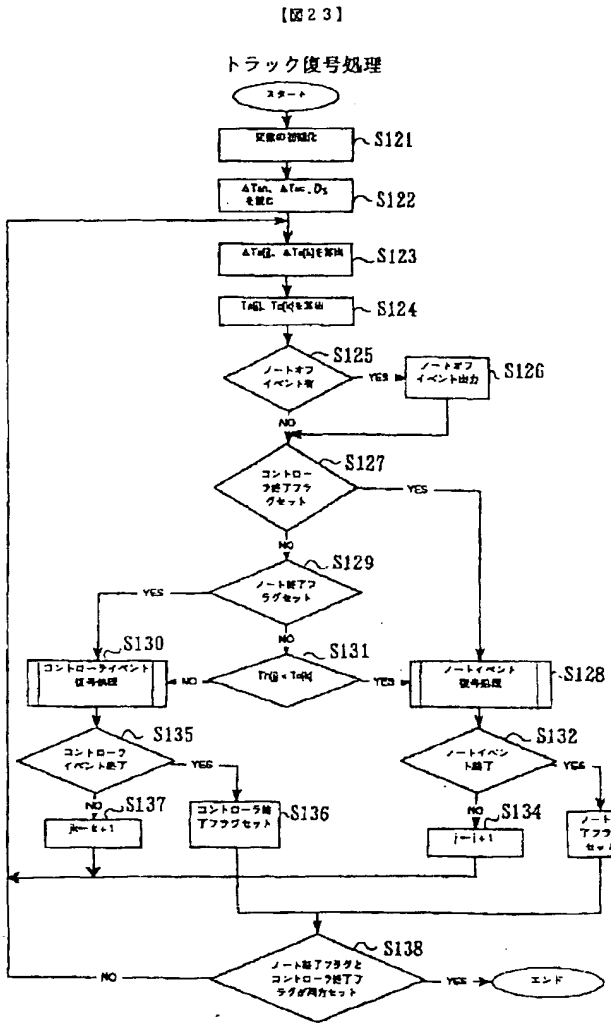
【図２４】



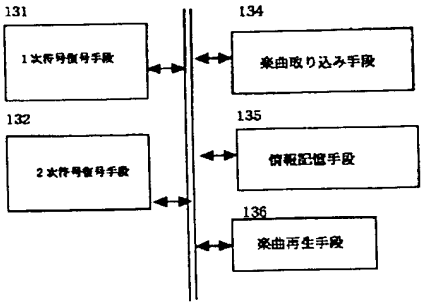
【図２９】



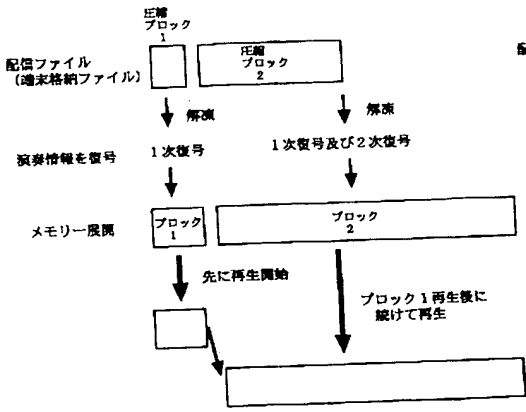
【図２３】



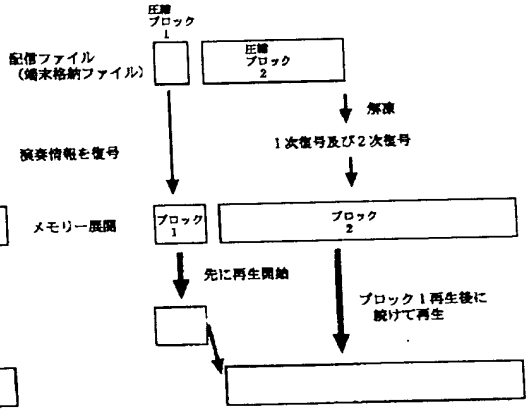
【図３１】



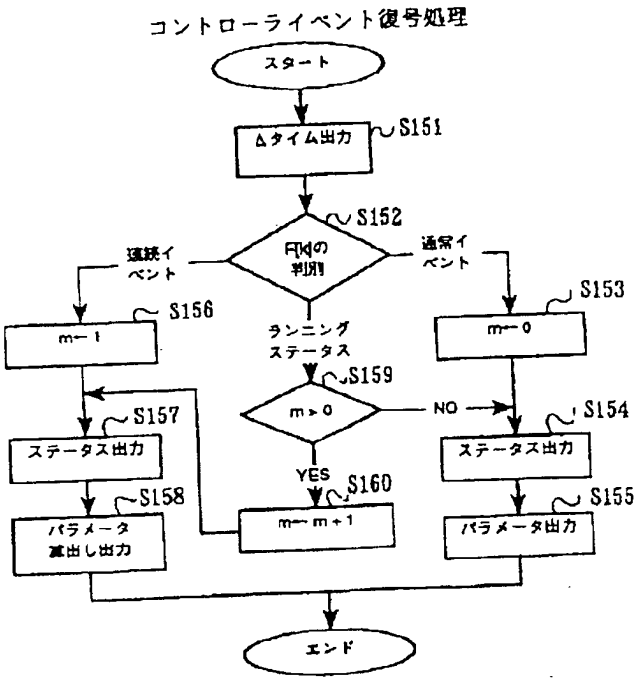
【図３２】



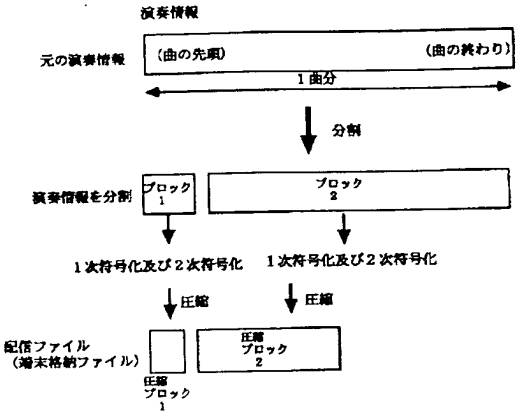
【図３３】



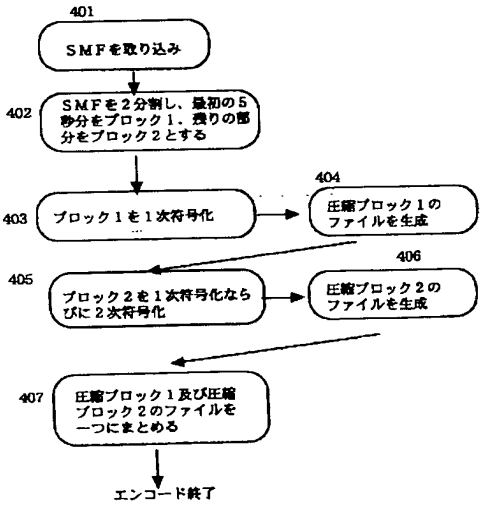
【図２７】



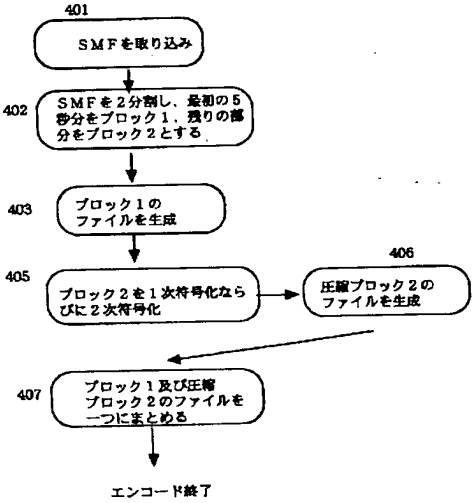
【図３４】



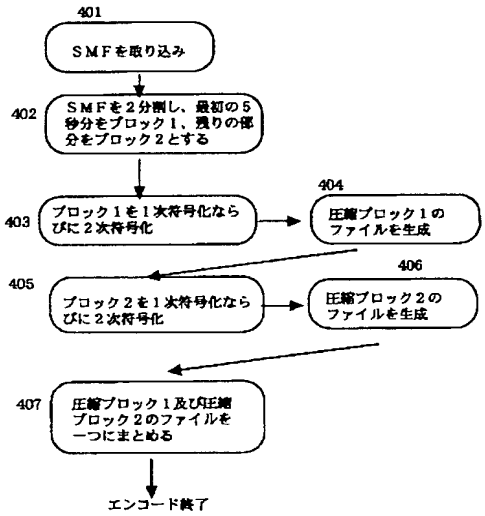
【図３５】



【図３６】



【図３７】



フロントページの続き

(51)Int.Cl.[7]	識別記号	Ｆ Ｉ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｈ０４Ｍ　1/00		Ｈ０４Ｍ　1/00	Ｂ